1. A method of mapping a plurality of tasks and data onto a multiple processor, distributed memory hardware architecture, the method comprising:

a) describing a task-level network of behaviors, each of the task-level network of behaviors being related through control and data flow;

b) predicting a schedule of tasks for the task-level network of behaviors; and

c) allocating the plurality of tasks and data to at least one of the multiple processors and to at least one of distributed memory, respectively, in response to the predicted schedule of tasks.

2. The method of claim 1 wherein the predicting the schedule of tasks comprises minimizing execution time of the plurality of tasks.

3. The method of claim 2 wherein the predicting the schedule of tasks comprises minimizing the schedule of tasks by allocating data to the distributed memories in order to minimize data transfers.

4. The method of claim 1 wherein the predicting the schedule of tasks comprises maximizing parallel execution of the plurality of tasks on at least two processors of the multiple processors.

5. The method of claim 1 wherein the allocating the plurality of tasks comprises allocating tasks to one of the multiple processors having optimal processor resources for the tasks.

6. The method of claim 1 wherein the predicting the schedule of tasks comprises

using a resource-based model of the architecture to predict the schedule of tasks.

7. The method of claim 1 wherein the predicting the schedule of tasks comprises using an interval graph and an execution time model of the task-level network of behaviors to predict the schedule of tasks.

8. The method of claim 1 wherein the allocating the plurality of tasks and data comprises an iterative allocation process.

9. The method of claim 8 wherein the iterative allocation process comprises using a demand-driven and constraint-based objective function.

10. The method of claim 1 wherein the describing a task-level network of behaviors comprises describing a task-level network of behaviors in a high-level programming language.

11. The method of claim 10 wherein the describing the task-level network of behaviors in the high-level programming language comprises parsing the high-level programming language into an intermediate form.

12. The method of claim 1 further comprising generating machine executable code for the multiple processor, distributed memory hardware architecture based at least in part on the allocating the plurality of tasks and data.

13. The method of claim 1 wherein the allocating the plurality of data to the distributed memories comprises allocating data to shared memories.

14. The method of claim 1 wherein the allocating the plurality of data to the

distributed memories comprises allocating data to private memories.

15. A method for generating a control graph for a compiler used to map a plurality of tasks and data onto a multiple processor, distributed memory hardware architecture, the method comprising:

a) parsing a plurality of tasks into an internal compiler form of interconnected task nodes; and

b) linking a compiler representation to each interconnected task node using directed edges for the purpose of substantially simultaneously mapping the tasks to multiple processors in the multiple processor, distributed memory hardware architecture.

16. The method of claim 15 further comprising binding the interconnected task nodes to the directed edges.

17. The method of claim 15 further comprising parsing a plurality of data blocks into an internal compiler form of data nodes and linking a compiler representation to each data node using directed edges.

18. The method of claim 15 wherein the directed edges represent time intervals.

19. The method of claim 18 wherein the time interval comprises the time period between tasks.

20. The method of claim 18 wherein the time interval comprises the time period from beginning a task to ending a task.

21. The method of claim 18 wherein the time interval comprises a set time period.

22. The method of claim 18 wherein the time interval comprises a time period between time periods.

23. A method for executing a schedule of tasks in a multiple processor, distributed memory architecture, the method comprising:

a) generating the schedule of tasks based at least in part on a task-level network of behaviors;

b) calculating a demand function based at least in part on a constraint related to at least one of a plurality of tasks in the schedule of tasks; and

c) allocating a task having highest priority to a processor having least cost according to the demand function.

24. The method of claim 23 further comprising allocating a data block to a memory in the distributed memory.

25. The method of claim 23 wherein the demand function is calculated based at least in part on the task-level network of behaviors.

26. The method of claim 23 wherein the demand function is calculated based at least in part on an impact on the schedule of tasks.

27. The method of claim 23 wherein the demand function is calculated based at least in part on an impact on data movement.

28. The method of claim 23 wherein the demand function is calculated based at least in part on prior allocation decisions.

29. The method of claim 23 wherein the cost is defined as the least negative impact on at least one performance factor.

30. The method of claim 29 wherein the at least one performance factor comprises the schedule of tasks.

31. The method of claim 29 wherein the at least one performance factor comprises data movement.

32. The method of claim 23 further comprising allocating a task having next highest priority to a processor having next least cost according to the demand function.

33. The method of claim 23 further comprising recalculating the demand function in response to each task in the plurality of tasks being allocated to a processor.

34. A compiler for mapping a plurality of tasks and data onto a multiple processor, distributed memory architecture, the compiler comprising:

a) means for describing a task-level network of behaviors, each of the task-level network of behaviors being interrelated through control and data flow dependencies;

b) means for predicting a schedule of tasks for the task-level network; and

c) means for allocating the plurality tasks and data to at least one of the multiple processors and at least one of the distributed memories,

respectively, in response to the predicted schedule of tasks.

35. The method of claim 34 further comprising means for producing machine executable code for the architecture based at least in part on the means for allocating the plurality of tasks and data.

36. The method of claim 34 wherein the multiple processors communicate using the distributed memories.